**ASSIGNMENT 8**

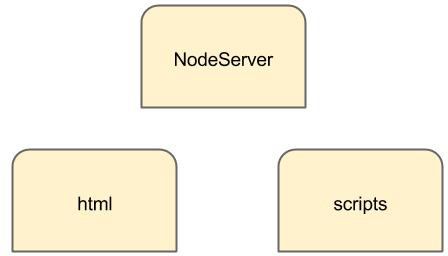
Your task is to write a new web server that will serve files from the file system.  You'll call it bestwebserver.js.

It will be similar to the second server we created in section 16.4: webserver.js.  However instead of the fixed HTML source document './html/nodedemo.html', your server will serve the resource requested by the user on the address bar as well as associated resources requested by the browser (such as JavaScript or CSS files and images).

For example when the user types: http://localhost:8080**/html/calculator.html**

Your server will serve the page from the source document .**/html/calculator.htm**l.

Your server will be stored in a folder such as NodeServer.  That folder includes two subfolders:  html and scripts.  Html documents such as calculator.html will be stored in the html subfolder.  Associated client-side JavaScript files such as add.js will be stored in the separate scripts subfolder.



You can access the full url entered by the user (or requested by the browser)  through the request object as request.url.

To parse it, you can **use the url module available in Node**.  Make sure you require it first (just like we required the http module in the webserver.js).

To get the resource name specified in request.url, you can then use:

var resource = url.parse(request.url).pathname;

The pathname is the portion of the url that comes after the hostname and port number and includes the initial forward slash /.

For the example above the pathname will be:  /html/calculator.html

Your server should serve the content of that file with the appropriate error handling in case that file does not exist.

Your server should serve HTML documents, JavaScript files, CSS files and gif images.  It has to specify the correct Content-Type for each in the header.  The Content-Type header for each of the resources above is as follows:

html files:  {'Content-Type': 'text/html; charset = UTF-8'}

JavaScript files (js): {'Content-Type': 'application/javascript'; charset = UTF-8'}

css files: {'Content-Type': 'text/css; charset = UTF-8'}

gif images: {'Content-Type': 'image/gif'})

For any other resource type, you may use {'Content-Type': 'text/plain; charset = UTF-8'}

You will **use the extension of the resource requested to determine its content type**.  You'll **use the path module** available in Node to get the extension of the resource specified.  Make sure you require it first  (just like we required the http module in the webserver.js)

To get the extension name for a given resource, you can then use path.extname(resource).

Note that the dot separator is included in the extension name so path.extname('./html/calculator.html') will return '.html'.

If no resource is requested (the user enters http://localhost:8080), the program should serve a default web page ./html/home.html.

If the requested resource does not exist, the program should issue an appropriate error message.

Start with the file webserver.js.

Modify it and upload your solution as bestwebserver.js.

Make sure you test it with valid and invalid file names before you submit it.

The screencast below demonstrates how you would test your new web server.  Note that all the resources are served through localhost NOT through the file scheme.



**Grading Rubric:**

The program uses the url module correctly to parse the requested url - 15 point

The program uses the path module correctly to get the corresponding extension - 15 points

The program serves HTML documents, JavaScript files, CSS files and gif images correctly - 20 points

The program serves a default web page when no specific resource is requested - 10 points

The program includes appropriate error handling when the resource requested does not exist - 10 points

**Answer**

* application/javascript[bestwebserver.js](https://myetudes.org/access/mneme/content/private/mneme/09ae2205-2717-4bfc-00cf-33f5bdcd7b48/submissions/15164104/ca016c57-16fc-4c6a-00cd-3fa2d0f5be31/bestwebserver.js)

[[https://myetudes.org/ambrosia_library/icons/collapse.gif](https://myetudes.org/portal/tool/acd42055-9bd4-4630-8071-c0425c2388c3/review/15164104/list) Model Answer](https://myetudes.org/portal/tool/acd42055-9bd4-4630-8071-c0425c2388c3/review/15164104/list)

// bestwebserver Solution

// The following function will be called whenever

// the server receives a request.

function servePage(request, response) {

  // Extract the filename from the request.

  var contentHeader;

  var filename = '.' + url.parse(request.url).pathname;

  // If the user does not enter a file name,

  // we serve the page home.html

  if (filename === './') {

    filename = './html/home.html';

  }

  switch (path.extname(filename)) {

    case '.html':

      contentHeader = {'Content-Type': 'text/html; charset = UTF-8'};

      break;

    case '.css':

      contentHeader = {'Content-Type': 'text/css; charset = UTF-8'}

      break;

    case '.js':

      contentHeader = {'Content-Type': 'application/javascript; charset = UTF-8'}

      break;

    case '.gif':

      contentHeader = {'Content-Type': 'image/gif'}

      break;

    default:

      contentHeader = {'Content-Type': 'text/plain; charset = UTF-8'}

  }

  // Read the file asynchronously

  fs.readFile(filename, function (err, content) {

    if (err) { // If there is an error, set the status code

      response.writeHead(404, {

        'Content-Type': 'text/plain; charset = UTF-8'

      });

      response.write(err.message); // Include the error message body

      response.write(' - The page requested is not found.');

      response.end(); // Done

    } else { // Otherwise, the file was read successfully.

      response.writeHead(200, // Set the status code

       contentHeader) // and the content header;

      response.write(content); // Send file contents as response body

      response.end();

    }

  });

}

// load the url module

var url = require('url');

// load the path module

var path = require('path');

// Load the file system module

var fs = require('fs');

// load the http module

var http = require('http');

// create a server object

var server = http.createServer(servePage);

server.listen(8080);

console.log('Server running at http://localhost:8080');